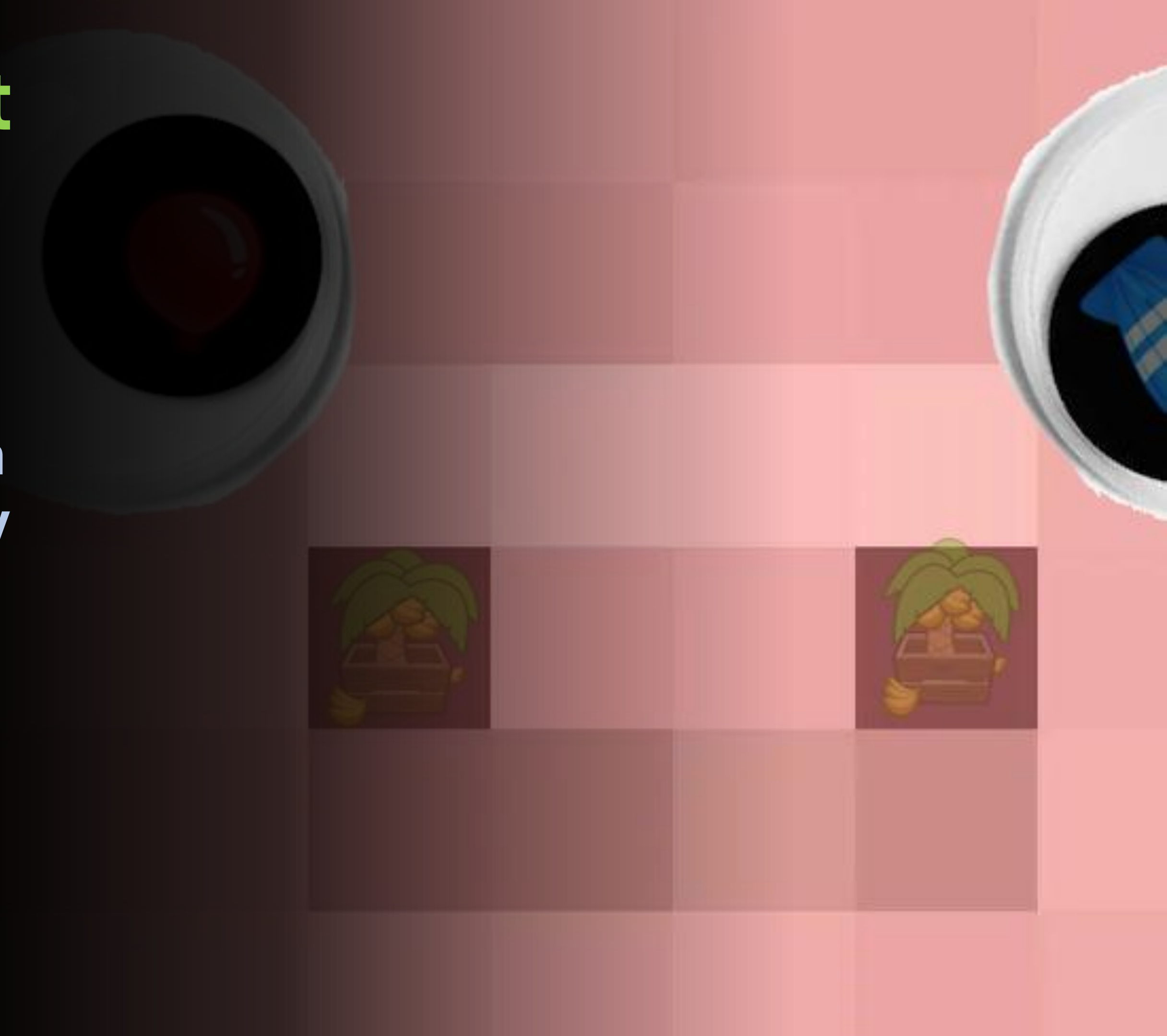
01-T2 Minecraft Game Mode

CS 4850 | Section 03
Spring 2025 | April 28th
Professor Sharon Perry

Ashley Ahn
Matthew Elledge
AnnaGrace Gwee
Bryan Nguyen
Logan Slicker



Preface

The KSU Minecraft server hosts many game modes and is looking for more: Thus, our game mode: **KSUBlocks Tower Defense!**

Our project's goal was to create a game within the Tower Defense Genre with inspired elements from BloonsTD6 and Sanctum 2, alongside our own takes and additions to it while fully supporting future development.

Overview

- **Requirements**
  - Milestones (1 | 2 | 3)
  - Functional
  - Non-Functional
- **Analysis & Design**
  - Tools & Dependencies
  - General Constraints
  - Architectural Drawings
  - Database Connection
- **Development**
  - Planning & GDD
  - Gameplay & Mechanics
  - Software Test Plan & Report (STP & STR)
- **Finalized Items**
  - GitHub Website
  - Completed Game

Requirements

Requirements - Milestones

Milestone 1

Draft the Game Design Document (GDD)

- Command structure & permissions defined
- Planned libraries
- Consider game mechanics, rules, and multiplayer function
- Plan on if the world or maps would be used and how
- General information believed necessary to the staff

GitHub Repository & IDEs

- Set up to begin coding after GDD approval

Milestone 2

Asynchronous Processes

- Verified that processes were handled off the main thread with an async function call
- Code is fully documented with explanations for code practices

Basic Prototype

- Basic model of the tower defense genre (functional towers and incoming enemies) showcased

Maps Designed

- Two fully functional maps and two rough draft maps were released

Milestone 3

Polish & User Friendly

- User Interfaces to maps, towers, and players added
- All four maps finalized with Hub zone added with leaderboard

Bug Testing and Playtesting

- Bugs removed and improved demo showcased at C-Day

Documentation

- All artifacts on GitHub Home Page
- Full documentation for players and admins on GitHub Wiki

Requirements - Functional

- **Map Selection:** The game is designed to handle map selection (through a GUI) and work with all four base-release maps alongside a hub zone (with a leaderboard) to send players to before and after games.
- **Round Management:** The game state is prepared upon map selection and allows the player to ready up to begin active gameplay (combat) or wait while player does passive gameplay (towers & upgrades). All functions the round calls are also asynchronous for overall server performance and functionality.
- **Mob Management:** Mobs are directed to follow the waypoints set to them. Upon reaching the end of their path, properly remove a player heart. Alternatively, upon defeat, mobs supply income to players, split if there is more than one player. Unique attributes, namely effect auras, are properly allocated to the right mobs.
- **Tower Management:** Towers can be built once players have enough coins to build them through the compass GUI. Towers can also be upgraded with their own GUI. Towers also properly provide income to the player who owns said tower and tracks mobs on the path accurately.
- **Player Management:** Players are capable of fighting mobs, upgrading their own stats, and building towers. Players are also supplied the necessary GUI calls and leaderboard updates for gameplay.

Requirements - Non-Functional

- **Capacity:** The game can properly run on any paper server, so long as they match the right Java version (1.21.4). This can be updated through modifying the plugin before rebuilding it, but currently is designed to support the latest Minecraft Java edition. The game should also be lightweight to handle higher rounds with increased and varying mob spawning alongside multiple player actions. This is accomplished through asynchronous code and utilizing the base game's architecture.

- **Usability:** The game needs to be easy to implement for developers and Minecraft servers while being easy to understand for both admins and players. As the Repo for the game is fully open source, developers can fully modify the game as needed. Documentation on the GitHub Wiki also ensures that both admins and players can read the information necessary to meet their experience requirements.

- **Modifiability:** The game should allow for full customization: This is completed through the config.yml allowing for full modification to any and all aspects of the base game. This includes stats, income, effects, basic additions and removal, and so on. Balance and new additions are easily added to the game with the functions allowing for modular code.

Analysis & Design

A&D – Tools & Dependencies

- **Plugins**
  - **Parties** – Creates the party system for multiplayer
  - **PlaceholderAPI** – Displays plugin information uniformly for servers
  - **SimpleScore** – Creates the leaderboard system for the game
- **APIs**
  - **Paper** – The platform for many modded Minecraft servers
  - **Parties**
  - **PlaceholderAPI**
- **IntelliJ MC Dev Plugin**
  - The basis for Minecraft Plugins

A&D – General Constraints

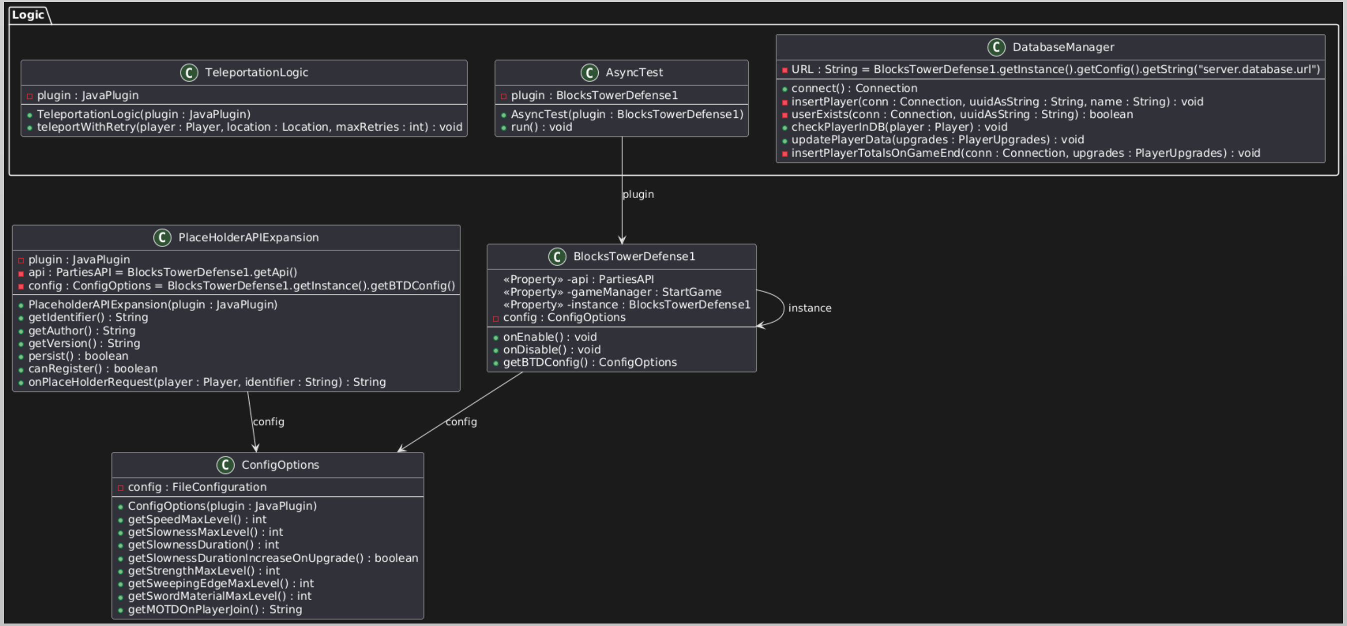
- **Asynchronous:** The game needs to utilize asynchronous tasks for saving/keeping data that is less important or time sensitive to ensure that the main processing thread is open and unblocked. Latency issues should be avoided.
- **Configurable:** The game, as stated before in non-functional requirements, should be configurable for future maintenance or changes as needed by server staff. This is accomplished with detailed documentation, code comment explanations, and easy to use functions.
- **Performance:** Of course, the game should run smoothly with no issues during gameplay. There should also be no issues once hosted on the KSU MC server with the other game modes. Code should be neat and efficient to highlight this.

A&D – Architectural Drawings

Please refer to the GitHub Page for more Architectural Drawings.

There are 6 total that are large enough to all take a slide while ensuring legibility.

As a sample, this is the General Logic.



TeleportationLogic

- □ plugin : JavaPlugin
- ● TeleportationLogic(plugin : JavaPlugin)
- ● teleportWithRetry(player : Player, location : Location, maxRetries : int) : void

AsyncTest

- □ plugin : BlocksTowerDefense1
- ● AsyncTest(plugin : BlocksTowerDefense1)
- ● run() : void

DatabaseManager

- ■ URL : String = BlocksTowerDefense1.getInstance().getConfig().getString("server.database.url")
- ● connect() : Connection
- ■ insertPlayer(conn : Connection, uuidAsString : String, name : String) : void
- ■ userExists(conn : Connection, uuidAsString : String) : boolean
- ● checkPlayerInDB(player : Player) : void
- ● updatePlayerData(upgrades : PlayerUpgrades) : void
- ■ insertPlayerTotalsOnGameEnd(conn : Connection, upgrades : PlayerUpgrades) : void

PlaceHolderAPIExpansion

- □ plugin : JavaPlugin
- ■ api : PartiesAPI = BlocksTowerDefense1.getApi()
- ■ config : ConfigOptions = BlocksTowerDefense1.getInstance().getBTDConfig()
- ● PlaceholderAPIExpansion(plugin : JavaPlugin)
- ● getIdentifier() : String
- ● getAuthor() : String
- ● getVersion() : String
- ● persist() : boolean
- ● canRegister() : boolean
- ● onPlaceHolderRequest(player : Player, identifier : String) : String

BlocksTowerDefense1

- «Property» -api : PartiesAPI
- «Property» -gameManager : StartGame
- «Property» -instance : BlocksTowerDefense1
- □ config : ConfigOptions
- ● onEnable() : void
- ● onDisable() : void
- ● getBTDConfig() : ConfigOptions

ConfigOptions

- □ config : FileConfiguration
- ● ConfigOptions(plugin : JavaPlugin)
- ● getSpeedMaxLevel() : int
- ● getSlownessMaxLevel() : int
- ● getSlownessDuration() : int
- ● getSlownessDurationIncreaseOnUpgrade() : boolean
- ● getStrengthMaxLevel() : int
- ● getSweepingEdgeMaxLevel() : int
- ● getSwordMaterialMaxLevel() : int
- ● getMOTDOnPlayerJoin() : String

A&D – Database Connection

SQLite

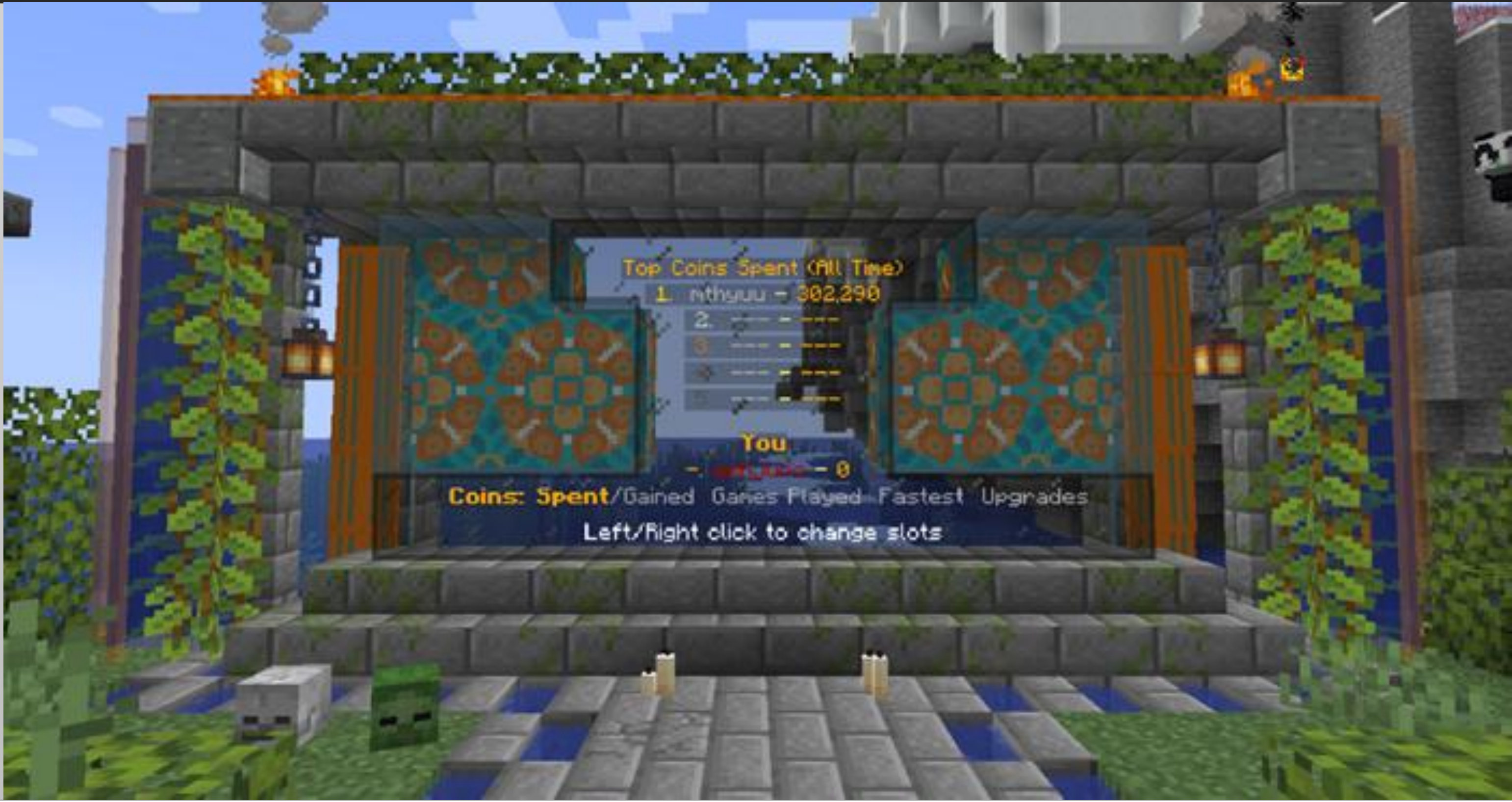
Lightweight, free, and easy to implement

Game Data

Collected for User Concurrency and Data Tracking for the game's leaderboard system

Leaderboards

- Gold Spent
- Gold Gained
- Games Played
- Fastest Completion
- Total Upgrades Bought



Top Coins Spent (All Time)

1. mthyuu - 302,290

2. --- - ---

3. --- - ---

4. --- - ---

5. --- - ---

You

- [illegible] - 0

Coins: Spent/Gained Games Played Fastest Upgrades
Left/Right click to change slots

Development

Development – Planning & GDD

- **Project Plan**
  - **Timeline:** Class & Sponsor tasks
  - **Time Approximation:** Gantt Chart
  - **Generation Information:** How to create the game
  - **Meetings:** Established team communication and sponsor meetings
  - **Version Control:** GitHub through the KSUBlocksTD Organization Account

- **Game Design Document**
  - **"Creating the project on paper"**
    - Detailed game mechanics and how they would function
    - Rough ideas designed for maps, mobs, players, and overall gameplay
    - Commands and architecture of the game laid out
    - Resources and constraints required for our design

Development – Gameplay & Mechanics

Gameplay

- Designed with the style of Bloons Tower Defense as the main source of inspiration for towers (from the Tower Defense genre)
- Contains elements of player combat elements similar to Sanctum 2.

Mechanics

- Players have limited hearts
- Hostile mobs enter the map, follow the path, and remove hearts from the player should they reach the end.
  - Upon being defeated, they instead supply coins.
- Players can set record scores to place themselves on the leaderboard

- Players can build towers to combat mobs
  - Tower upgrades are generalized and focus on playstyles
  - Player combat can be expanded with upgrades and consumable items
- Multiple maps can be selected with varying design elements and playstyles

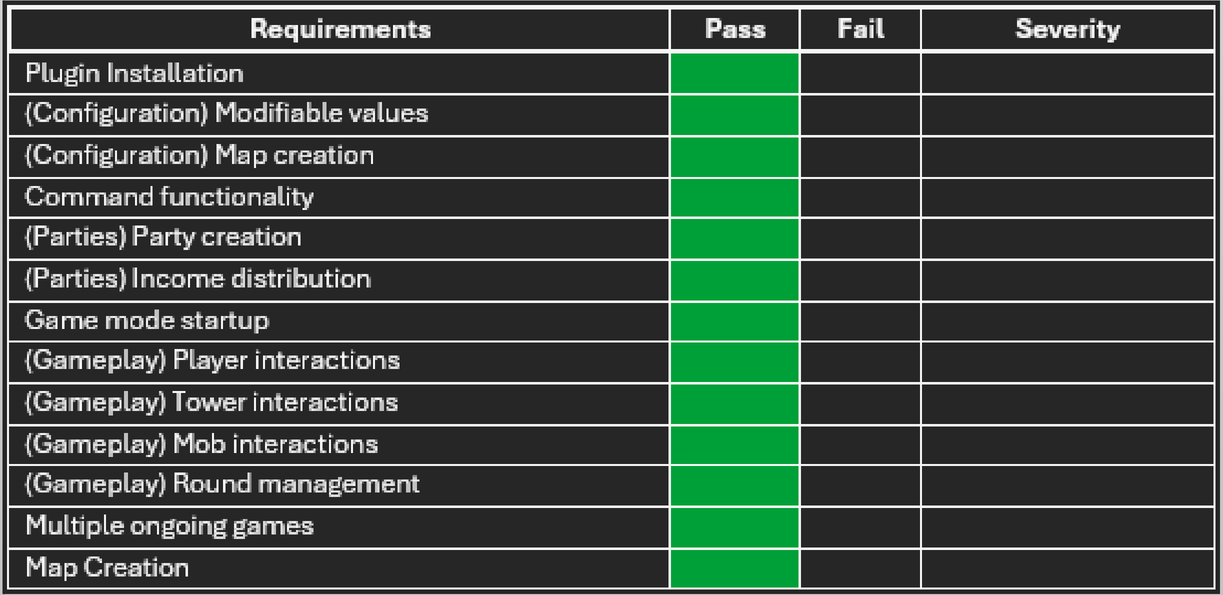
Development – STP & STR

Software Test Plan

- How the project will be tested
- Who will test what parts
- What aspects are to be tested,
- Where it will be tested.

The project was tested locally with all members of the team utilizing the testing table as seen.

As of project finalization, there were no issues found.

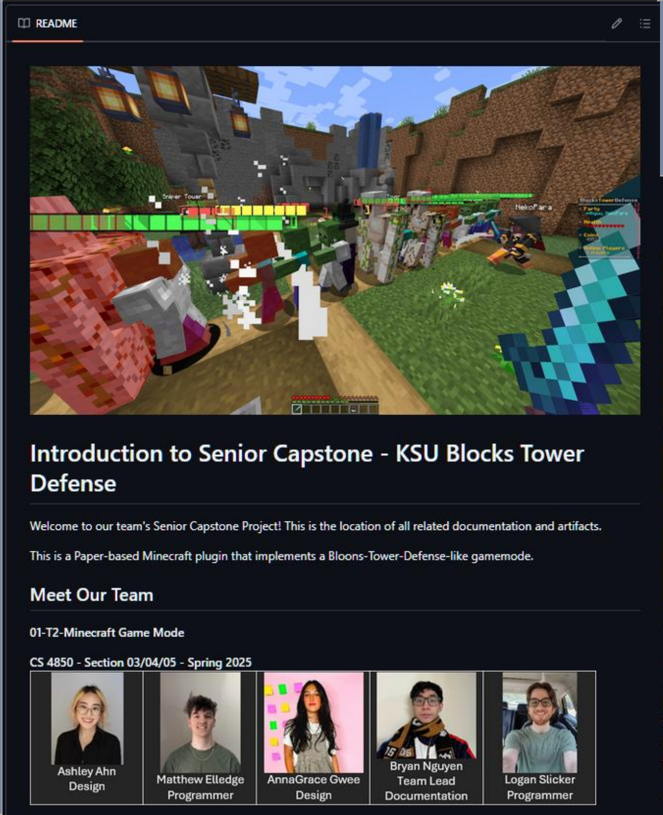


Requirements	Pass	Fail	Severity
Plugin Installation	🟩		
(Configuration) Modifiable values	🟩		
(Configuration) Map creation	🟩		
Command functionality	🟩		
(Parties) Party creation	🟩		
(Parties) Income distribution	🟩		
Game mode startup	🟩		
(Gameplay) Player interactions	🟩		
(Gameplay) Tower interactions	🟩		
(Gameplay) Mob interactions	🟩		
(Gameplay) Round management	🟩		
Multiple ongoing games	🟩		
Map Creation	🟩		

Finalized Items

Finalized Items – GitHub Website

Our GitHub Home Page contains basic information on our project alongside all artifacts throughout the semester



Introduction to Senior Capstone - KSU Blocks Tower Defense

Welcome to our team's Senior Capstone Project! This is the location of all related documentation and artifacts.

This is a Paper-based Minecraft plugin that implements a Bloons-Tower-Defense-like gamemode.

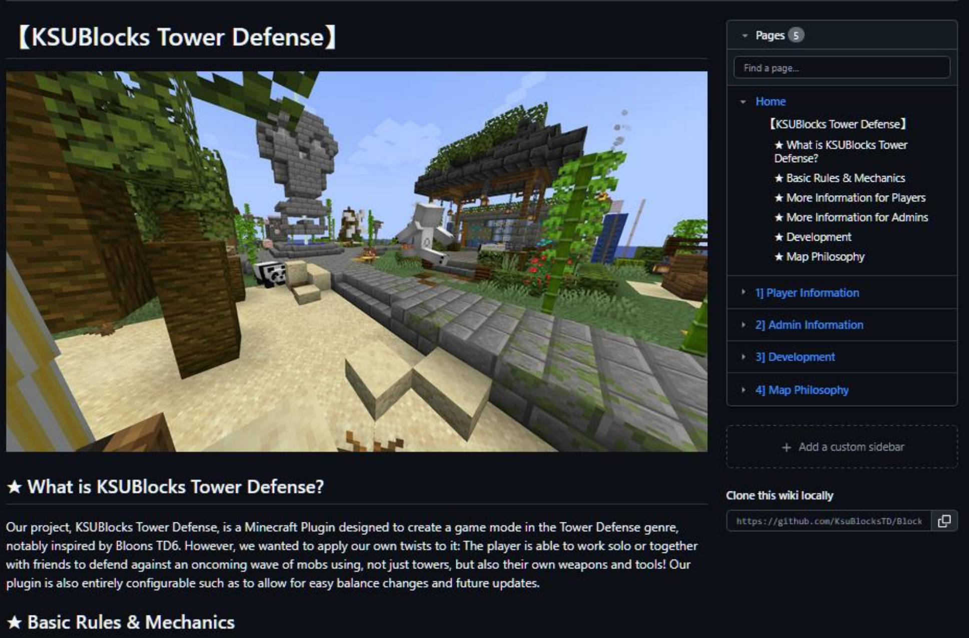
Meet Our Team

01-T2-Minecraft Game Mode

CS 4850 - Section 03/04/05 - Spring 2025

Ashley Ahn Design	Matthew Elledge Programmer	AnnaGrace Gwee Design	Bryan Nguyen Team Lead Documentation	Logan Slicker Programmer

Our GitHub Wiki Page contains a home page explaining the game alongside all documentation pertaining to players, admins, development, and map design.


Pages 5
Find a page...
Home
【KSUBlocks Tower Defense】
★ What is KSUBlocks Tower Defense?
★ Basic Rules & Mechanics
★ More Information for Players
★ More Information for Admins
★ Development
★ Map Philosophy
1] Player Information
2] Admin Information
3] Development
4] Map Philosophy
+ Add a custom sidebar
Clone this wiki locally
https://github.com/KsuBlocksTD/Block

【KSUBlocks Tower Defense】

★ What is KSUBlocks Tower Defense?

Our project, KSUBlocks Tower Defense, is a Minecraft Plugin designed to create a game mode in the Tower Defense genre, notably inspired by Bloons TD6. However, we wanted to apply our own twists to it: The player is able to work solo or together with friends to defend against an oncoming wave of mobs using, not just towers, but also their own weapons and tools! Our plugin is also entirely configurable such as to allow for easy balance changes and future updates.

★ Basic Rules & Mechanics

Finalized Items – Completed Game

Video Not Present for GitHub’s File Size Limit

Please Refer to the Youtube Video to See the Game Showcase

Conclusion

Through the usage of IntelliJ as our IDE alongside our plugins and APIs (such as Paper), we believe that we accomplished our goal in implementing a fun and functional Tower Defense game that stands out in its genre and utilizes the Minecraft assets and functions well. We hope our project has longevity on the KSU Minecraft server and receives regular updates!

Discord Link to the KSU Minecraft Server:
https://discord.gg/7xWVtaDvzK